



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/776,328	02/02/2001	Tim Scott Foster -	P4756 US	9582
22434	7590	08/26/2004	EXAMINER	
BEYER WEAVER & THOMAS LLP P.O. BOX 778 BERKELEY, CA 94704-0778			SHRADER, LAWRENCE J	
			ART UNIT	PAPER NUMBER
			2124	

DATE MAILED: 08/26/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/776,328	Applicant(s) FOSTER ET AL.	
	Examiner Lawrence Shrader	Art Unit 2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
 - If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
 - If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
 - Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 10 May 2004.
- 2a) ☒ This action is **FINAL**. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-37 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-37 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☒ All b) ☐ Some * c) ☐ None of:
1. ☒ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. _____.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) * | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

1. This office action is in response to the Applicant's amendment dated 5/10/2004.
2. The change of power of attorney submitted on 5/10/2004 is acknowledged.
3. Of the original claims 1 – 35, claims 1, 13, 14, 20, 22, 23, 31, and 32 have been amended; and new claims 36 and 37 have been added as requested by the Applicant in the amendment of 5/10/2004. Claims 1 – 37 have been rejected in this office action. The Applicant's arguments have been fully considered, but are moot in view of the new grounds of rejection made necessary by the amendments.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

5. Claim Claims 1, 2, 8, 11, 12; 13 – 15, 19; 20, 21; 22; 23, 24, 28; 31, 33; 36 and 37 are rejected under 35 U.S.C. 103(a) as being unpatentable over Logan, U.S. Patent 6,601,018 in view of Donohue, U.S. Patent 6,202,207.

In regard to claim 1:

“at least one test module configured to use the data entries of the file list to test at least one parameter of the software package;”

Logan discloses the testing of software components (column 2, lines 19 – 29), but does not explicitly disclose use of data from a file list to test a parameter of a software package. However, Donohue discloses a file list containing parameter information used to test pre-requisite components of a software package (column 16, lines 7 – 26). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine software component testing as taught by Logan with the pre-requisite component testing of Donohue because the combination allows the updater to discover problems before the software is run on the system by responding to external signals as taught by Donohue at column 16, lines 27 – 46.

“a framework operable to identify at least one test module defining a test of at least one parameter of the at least one software component of the package;”

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components (column 2, lines 19 – 29).

“a control module operable to access the framework to cause the at least one test module identified therein to perform the test defined thereby for verifying the package.”

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module, which accesses the client framework (column 4, lines 40 – 42; column 31 – 39) to perform a test.

In regard to claim 2, incorporating the rejection of claim 1:

“...wherein the framework identifies a plurality of test modules.”

Logan discloses a test suite in an automatic test framework (column 2, lines 19 50).

Art Unit: 2124

In regard to claim 8, incorporating the rejection of claim 2:

“...wherein the framework comprises a directory having a plurality of entries, each entry identifying one of the plurality of test modules.”

Logan discloses a repository for a plurality of test suites allowing the tester to know where to find (a directory) a specific test suite (column 7, lines 3 – 25).

In regard to claim 11, incorporating the rejection of claim 2:

“...wherein each of the plurality of test modules is formed by a script and the framework identifies each of the test modules by a name for the script.”

Logan teaches the plurality of test modules formed by a script, which is called by name (see column 5 code example).

In regard to claim 12, incorporating the rejection of claim 2:

“...wherein each of the test modules is formed by a software object.”

A test case (module) is formed by a software object as shown in column 5 of Logan.

In regard to claims 36 and 37, incorporating the rejection of claims 1 and 36 respectively:

“A software package verification tool as in claim 1 wherein the software package is compliant with the SOLARIS standard.”

“A software package verification tool as in claim 36 wherein the file list comprises a “pkgmap” file.”

Logan discloses a test suite in an automatic test framework (column 2, lines 19 50) that runs in a number of distributed environments (column 3, line 28 to column 4, line 27), but does not explicitly disclose the SOLARIS, a UNIX-based distributed environment standard.

However, Donohue discloses compatibility with a UNIX based operating system (column 8, lines 31 – 33), at least suggesting a compatibility with the UNIX based SOLARIS standard.

Therefore, it would have been obvious to one skilled in the art at the time the invention was

Art Unit: 2124

made to combine the distributed test environment taught by Logan with the UNIX based distributed system of Donohue, including the porting of a “pkgmap” being a package contents description file in a UNIX-based system, because the Donohue invention provides the means to register components from other networks as taught at column 8, lines 19 – 34 thus extending the integrated test environment of Logan according to a known standard.

In regard to claim 13:

a) “providing at least one test module configured to use the data entries of the file list to test at least one parameter of the software package;”

Logan discloses the testing of software components (column 2, lines 19 – 29), but does not explicitly disclose use of data from a file list to test a parameter of a software package. However, Donohue discloses a file list containing parameter information used to test pre-requisite components of a software package (column 16, lines 7 – 26). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine software component testing as taught by Logan with the pre-requisite component testing of Donohue because the combination allows the updater to discover problems before the software is run on the system by responding to external signals as taught by Donohue at column 16, lines 27 – 46.

“b) forming a framework operable to identify at least one test module for testing at least one parameter of the at least one software component of the package;”

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components (column 2, lines 19 – 29).

“c) forming a control module operable to access the framework to cause the at least one test module identified therein to perform the test for verifying the package.”

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module, which accesses the client framework (column 4, lines 40 – 42; column 31 – 39).

In regard to claim 14:

“a) providing a framework for identifying at least one test module, each said test module defining a test of at least one parameter of the at least one software component of the package wherein the test is configured to use the data entries of the file list to test the at least one parameter of the software package;”

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components (column 2, lines 19 – 29), but does not explicitly disclose use of data from a file list to test a parameter of a software package. However, Donohue discloses a file list containing parameter information used to test pre-requisite components of a software package (column 16, lines 7 – 26). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine software component testing as taught by Logan with the pre-requisite component testing of Donohue because the combination allows the updater to discover problems before the software is run on the system by responding to external signals as taught by Donohue at column 16, lines 27 – 46.

“b) accessing the framework to identify the at least one test module;”

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module, which accesses the client framework (column 4, lines 40 – 42; column 31 – 39).

“c) causing the at least one test module to perform the test defined thereby on the package.”

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module causing the initiation of a test on the client framework (column 4, lines 40 – 42; column 31 – 39).

In regard to claim 15, incorporating the rejection of claim 14:

“...wherein the framework identifies a plurality of the test modules.”

Logan discloses a test suite in an automatic test framework (column 2, lines 19 50).

In regard to claim 19, incorporating the rejection of claim 15:

“...providing a directory in the framework, wherein the directory has a plurality of entries, each entry identifying one of the plurality of test modules.”

Logan discloses a repository for a plurality of test suites allowing the tester to know where to find (a directory) a specific test suite (column 7, lines 3 – 25).

In regard to claim 20:

“a) “at least one test module configured to use the data entries of the file list to test at least one parameter of the software package;”

Logan discloses the testing of software components (column 2, lines 19 – 29), but does not explicitly disclose use of data from a file list to test a parameter of a software package. However, Donohue discloses a file list containing parameter information used to test pre-requisite components of a software package (column 16, lines 7 – 26). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine software component testing as taught by Logan with the pre-requisite component testing of Donohue because the combination allows the updater to discover problems before the software is run on the system by responding to external signals as taught by Donohue at column 16, lines 27 – 46.

“b) a framework to identify at least one test module;”

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components (column 2, lines 19 – 29).

“c) a control module operable to access the framework for causing the at least one test module identified therein to perform the test defined thereby for verifying the package.”

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module, which accesses the client framework (column 4, lines 40 – 42; column 31 – 39).

In regard to claim 21, incorporating the rejection of claim 20:

“...wherein the system comprises a computer including a processor, memory and software held in the memory and operable to control the processor, the software forming: said framework and said control module.”

Logan discloses a framework operating on computer system having a processor, memory, and software held in memory (column 3, lines 28 – 64).

In regard to claim 22:

“a) a memory for storing software;”

Logan discloses a framework operating on computer system having a processor, memory, and software held in memory (column 3, lines 28 – 64).

*“b) a processing unit for executing the software to carry out the steps of
(i) providing a framework to identify at least one test module defining a
test of at least one parameter of the at least one software component of
the package; and*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components (column 2, lines 19 – 29).

(ii) providing a control module operable to access the framework for causing the at least one test module identified therein to perform a test that uses the data entries of the file list to test the at least one parameter of the software package thereby verifying the package.”

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module, which accesses the client framework (column 4, lines 40 – 42; column 31 – 39), but does not explicitly disclose use of data from a file list to test a parameter of a software package. However, Donohue discloses a file list containing parameter information used to test pre-requisite components of a software package (column 16, lines 7 – 26). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine software component testing as taught by Logan with the pre-requisite component testing of Donohue because the combination allows the updater to discover problems before the software is run on the system by responding to external signals as taught by Donohue at column 16, lines 27 – 46.

In regard to claim 23:

“a) providing a framework for identifying at least one test module, each said test module configured to use the data entries of the file list to test at least one parameter of the software package thereby defining a test of at least one parameter of the at least one software component of the package;”

Logan discloses the testing of software components (column 2, lines 19 – 29), but does not explicitly disclose use of data from a file list to test a parameter of a software package. However, Donohue discloses a file list containing parameter information used to test pre-requisite components of a software package (column 16, lines 7 – 26). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine software

Art Unit: 2124

component testing as taught by Logan with the pre-requisite component testing of Donohue because the combination allows the updater to discover problems before the software is run on the system by responding to external signals as taught by Donohue at column 16, lines 27 – 46.

“b) accessing the framework to identify the at least one test module;”

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module, which accesses the client framework (column 4, lines 40 – 42; column 31 – 39).

“c) causing the at least one test module to perform the test defined thereby on the package.”

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module causing the initiation of a test on the client framework (column 4, lines 40 – 42; column 31 – 39).

In regard to claim 24, incorporating the rejection of claim 23:

“...wherein the framework identifies a plurality of test modules.”

Logan discloses a test suite in an automatic test framework (column 2, lines 19 50).

In regard to claim 28, incorporating the rejection of claim 24:

“...providing a directory in the framework, wherein the directory has a plurality of entries, each entry identifying one of the test modules.”

Logan discloses a repository for a plurality of test suites allowing the tester to know where to find (a directory) a specific test suite (column 7, lines 3 – 25).

In regard to claim 31:

“a) receiving the software package wherein the software package includes a file list having data entries associated with parameters for the at least one software component;”

Logan teaches that a package is received to perform a test (column 4, lines 36 – 54), but does not explicitly disclose use of data from a file list to test a parameter of a software package. However, Donohue discloses a file list containing parameter information used to test pre-requisite components of a software package (column 16, lines 7 – 26). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine software component testing of a software package as taught by Logan with the pre-requisite component testing of Donohue because the combination allows the updater to discover problems before the software is run on the system by responding to external signals as taught by Donohue at column 16, lines 27 – 46.

“b) accessing a framework that references at least one test module to identify the at least one test module from the framework, each said test module configured to use the data entries of the file list to define a test of the software package;”

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components (column 2, lines 19 – 29), but does not explicitly disclose use of data from a file list to test a parameter of a software package. However, Donohue discloses a file list containing parameter information used to test pre-requisite components of a software package (column 16, lines 7 – 26). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine software component testing as taught by Logan with the pre-requisite component testing of Donohue because the combination allows the updater to discover problems before the software is run on the system by responding to external signals as taught by Donohue at column 16, lines 27 – 46.

“c) performing the test defined by the at least one test module on the package.”

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module, which accesses the client framework (column 4, lines 40 – 42; column 31 – 39) to perform a test.

6. Claims 3 – 7, 9, 10; 16 – 18; 25 – 27, 29, 30, 32, 34, and 35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Logan, U.S. Patent 6,601,018, in view of Donohue, U.S. Patent 6,202,207, and further in view of Mastronardi, U.S. Patent 6,346,951.

In regard to claim 3, incorporating the rejection of claim 2:

“...wherein the framework identifies a priority for each of the test modules.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

In regard to claim 4, incorporating the rejection of claim 3:

“...wherein the control module is operable to cause the test modules to be executed sequentially according to the priority identified in the framework for each of the test modules.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

In regard to claim 5, incorporating the rejection of claim 1:

“...wherein a mechanism is provided for identifying the at least one test module as being one of active and not active.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active or not. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a

Art Unit: 2124

means to determine if certain tasks or modules are active or not active as taught by Mastronardi (column 8, lines 25 – 39).

In regard to claim 6, incorporating the rejection of claim 5:

“...wherein the mechanism for identifying the at least one test modules as being one of active and not active is included in the framework.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active or not. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active or not active as taught by Mastronardi (column 8, lines 25 – 39).

In regard to claim 7, incorporating the rejection of claim 5:

“...wherein the mechanism for identifying the at least one test modules as being one of active and not active is included in the control module.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active or not. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test

Art Unit: 2124

framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active or not active as taught by Mastronardi (column 8, lines 25 – 39).

In regard to claim 9, incorporating the rejection of claim 8:

“...wherein each entry defines a priority for the one of the test modules identified therein.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

In regard to claim 10, incorporating the rejection of claim 8:

“...wherein the identity of the one of the test modules defines its priority.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) based on the identity of the test in order to determine if a certain task is active. Therefore, it would have

Art Unit: 2124

been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

In regard to claim 16, incorporating the rejection of claim 15:

“...wherein a priority for each of the test modules is identified in the framework.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

In regard to claim 17, incorporating the rejection of claim 15

“...sequentially causing each of the test modules to be executed according to the priority identified for each of the test modules.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a

Art Unit: 2124

supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

In regard to claim 18, incorporating the rejection of claim 15:

“...identifying each of the test modules as being one of active and not active.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active or not. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active or not active as taught by Mastronardi (column 8, lines 25 – 39).

In regard to claim 25 incorporating the rejection of claim 24:

“...wherein a priority for each of the test modules is identified in the framework.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a

Art Unit: 2124

supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

In regard to claim 26, incorporating the rejection of claim 25:

“...sequentially causing the test modules to be executed according to the priority identified for each of the test modules.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

In regard to claim 27, incorporating the rejection of claim 24:

“...identifying each of the test modules as being one of active and not active.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active or not. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active or not active as taught by Mastronardi (column 8, lines 25 – 39).

In regard to claim 29, incorporating the rejection of claim 28:

“... wherein each entry defines a priority of the test module identified thereby.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

In regard to claim 30, incorporating the rejection of claim 28:

“... wherein the identity of a module defines its priority.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) based on the identity of the test in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

In regard to claim 32, incorporating the rejection of claim 31:

“...including repeating steps (b) and (c) to perform a sequence of tests, the order in which the tests are performed being determined by relative priorities assigned to each of the at least test module.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

In regard to claim 34, incorporating the rejection of claim 33:

“...wherein the data structure further comprises a second data field identifying a priority for each of the test modules represented by the data in the first data field, the priority defining an order of execution of the test modules.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

In regard to claim 35 incorporating the rejection of claim 33:

“...wherein the data structure further comprises a third data field identifying the one of a plurality of test modules represented by the data in the first data field as being one of active and not active.”

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Donohue discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active or not. Therefore, it would have been obvious to one skilled

in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the teaching of Donohue providing a file list of parameters, further combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active or not active as taught by Mastronardi (column 8, lines 25 – 39).

Conclusion

7. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire **THREE MONTHS** from the mailing date of this action. In the event a first reply is filed within **TWO MONTHS** of the mailing date of this final action and the advisory action is not mailed until after the end of the **THREE-MONTH** shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than **SIX MONTHS** from the date of this final action.

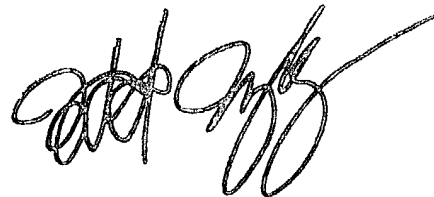
8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Lawrence Shrader whose telephone number is (703) 305-8046. The examiner can normally be reached on M-F 08:00-16:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703) 305-9662. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Lawrence Shrader
Examiner
Art Unit 2124

August 17, 2004

A handwritten signature in black ink, appearing to read 'Todd Ingberg', with a long horizontal stroke extending to the right.

TODD INGBERG
PRIMARY EXAMINER